

ΕΡΓΑΣΙΑ 2^η

Ημερομηνία Παράδοσης: 06-01-2021

Φοιτητής: ΓΙΑΝΝΑΚΟΣΙΑΝ ΑΝΕΣΤΗΣ

Τμήμα: ΠΛΣ50-ΗΛΕ45

Επιβλέπων Καθηγητής: Κος ΜΑΥΡΟΜΜΑΤΗΣ ΓΕΩΡΓΙΟΣ

Πρόλογος

Πριν ξεκινήσει η επεξήγηση των επιμέρους θεμάτων της Εργασίας 2, να υπενθυμίσω πως τα προγράμματα δημιουργήθηκαν στο **IntelliJ IDEA** της εταιρείας JETBRAINS, με JDK version 15, αλλά δοκιμάστηκαν και στο **Bluej** για την εύρυθμη λειτουργία τους. Επίσης, διατήρησα αντίστοιχη δομή κειμένου με το explain.doc της Εργασίας 1, καθώς σύμφωνα με την αξιολόγησή σας, βρήκατε το συνοδευτικό αρχείο ικανοποιητικό.

Τέλος, να τονίσω πως στο **Θέμα 1** της Εργασίας 2 και ύστερα από την διευκρίνιση σας περί καταχώρισης διαδρομής αρχείου από τον χρήστη, δημιούργησα κάποιες επιπλέον μεθόδους παρότι δεν ήταν υποχρεωτικό, απλά έκρινα καλό, την κατανομή κάποιων λειτουργιών. Κυρίως το έκανα, για να επιτευχθεί σε μεγαλύτερο βαθμό το single responsibility, να είναι πιο ευανάγνωστος και κατανοητός ο κώδικας και για να προσθέσω validation στις καταχωρίσεις του user.



Θέμα 1: Επεξεργασία Αρχείου με URLs

Ένας προγραμματιστής θέλει να εξάγει από ένα αρχείο κειμένου (ενδεικτικά μαζί με την άσκηση δίνεται το αρχείο **mySources.src**) που περιέχει διευθύνσεις ιστοσελίδων, τις διευθύνσεις των ιστότοπων χωρίς περαιτέρω εξειδικεύσεις όπως συγκεκριμένες σελίδες εντός του ιστότοπου ή άλλα ερωτήματα. Δηλαδή αν σε μια γραμμή του αρχείου περιέχεται η διεύθυνση **www.w3schools.com/quiztest/quiztest.asp?qtest=JAVA** τότε ο προγραμματιστής θέλει να εξάγει το τμήμα **www.w3schools.com**

Να γράψετε μια κλάση **URLsVisited** η οποία έχει ως σκοπό να διαβάζει από ένα αρχείο που περιέχει διευθύνσεις ιστοσελίδων μία μία τις γραμμές και να απομονώνει τους κόμβους που αυτές περιέχουν. Αυτούς θα τους αποθηκεύει σε μια δομή ArrayList και θα τους τυπώνει στην οθόνη. Στη συνέχεια θα εντοπίζει τους μοναδικούς κόμβους και θα τους τυπώνει στην οθόνη μετά από κατάλληλο μήνυμα.

Η κλάση θα περιέχει τα παρακάτω μέλη:

- Ένα ιδιωτικό (private) μονοδιάστατο ArrayList αλφαριθμητικών με όνομα **myURLs**.
- Έναν κατασκευαστή ο οποίος θα δέχεται ως όρισμα το όνομα ενός αρχείου κειμένου (που περιέχει διευθύνσεις ιστοσελίδων). Ο κατασκευαστής θα διαβάζει, γραμμή-γραμμή, τα περιεχόμενα του αρχείου, θα διαχωρίζει τα μέρη της διεύθυνσης της κάθε ιστοσελίδας και θα αποθηκεύει τη διεύθυνση του κόμβου που αυτή περιέχει στο ArrayList **MyURLs**.
- Μια δημόσια μέθοδο με όνομα **removeDuplicates** που δε θα έχει κάποιο όρισμα. Αυτή αρχικά θα αφαιρεί τα διπλότυπα στοιχεία από το ArrayList **myURLs** και στη συνέχεια θα το ταξινομεί.
- Μια δημόσια μέθοδο **printAll** που θα τυπώνει όλα τα στοιχεία που περιέχει το ArrayList **myURLs**.

Για να δοκιμάσετε την κλάση σας δημιουργήστε μια δεύτερη κλάση με όνομα **URLsVisitedApp** που θα περιέχει τη μέθοδο **main** και η οποία θα δέχεται από τον χρήστη το όνομα του αρχείου που περιέχει τις διευθύνσεις των ιστοσελίδων (ενδεικτικά μαζί με την άσκηση δίνεται το αρχείο **mySources.src**). Για διευκόλυνση σας μπορείτε να δίνετε από το πληκτρολόγιο ολόκληρο το μονοπάτι μαζί με το όνομα του αρχείου.

Η κλάση **URLsVisitedApp** θα περιέχει επιπλέον και μία μέθοδο **fileAnalyzer** που θα δέχεται ως όρισμα το όνομα του αρχείου που έδωσε ο χρήστης. Αυτή θα υλοποιεί τα παρακάτω:

- Θα δημιουργεί ένα αντικείμενο **URLsVisited** με όρισμα αυτό το αρχείο.
- Θα καλεί τη μέθοδο στιγμιοτύπου **printAll** προκειμένου να τυπώσει όλες τις διευθύνσεις των κόμβων που περιέχονται στο αρχείο.
- Θα καλεί τη μέθοδο στιγμιοτύπου **removeDuplicates** προκειμένου να εντοπίσει τις μοναδικές διευθύνσεις των κόμβων και να τις ταξινομήσει.
- Θα καλεί ξανά τη μέθοδο στιγμιοτύπου **printAll** προκειμένου να τυπώσει τις μοναδικές διευθύνσεις των κόμβων που περιέχονται στο αρχείο.

Για την επίλυση της άσκησης δημιουργήθηκαν δύο κλάσεις η **public class URLsVisited** (*URLsVisited.java*) και η **public class URLsVisitedApp** (*URLsVisitedApp.java*) στην οποία βρίσκεται η **main()** μέθοδος.

➤ Υλοποίηση URLsVisited.java

Έγινε import των α) **java.io.FileNotFoundException** για να επιτευχθεί ο χειρισμός της συγκεκριμένης εξαίρεσης, β) **java.io.File** για να αναγνωρίζεται ο αντίστοιχος τύπος παραμέτρου, κλάσης *File*, γ) **java.util.*** για να χρησιμοποιήσουμε τις κλάσεις ***Scanner***, ***ArrayList*** και ***TreeSet*** και να δημιουργήσουμε αντικείμενα αυτών.

A) Μεταβλητές Κλάσης

Οι 3 μεταβλητές που δηλώθηκαν είναι οι εξής:

- **private** *Scanner fileReader*, στην οποία θα αποθηκεύσουμε τα περιεχόμενα που θα διαβαστούν από το αρχείο που θα δοθεί ως παράμετρος.
- **private** *ArrayList<String> myURLs = new ArrayList<>()*, η οποία είναι ένας μονοδιάστατος πίνακας για την αποθήκευση των host names των urls που βρίσκονται στο αρχείο που θα δίνεται στην **public URLsVisited** (*File filename*). π.χ για το url www.test.gr/just_testing το host name αντιστοιχεί στο www.test.gr.
- **private** *boolean processedArraylist*, την οποία θα εκμεταλλευτώ στην μέθοδο **public void removeDuplicates()** για να γνωρίζει το πρόγραμμα εάν το arraylist που πρόκειται να προσπελαστεί έχει υποστεί μετατροπή από την εν λόγω μέθοδο.

B) Ανάλυση Κατασκευαστή

Η κλάση αποτελείται από 1 κατασκευαστή, ως εξής:

- **public URLsVisited(String filepath)**. Ο κατασκευαστής της κλάσης, παίρνει ως όρισμα τη διαδρομή αρχείου που δίνει ο χρήστης και δημιουργεί ένα αντικείμενο της *File* με όνομα **filename**. Διαβάζει το αρχείο που βρίσκεται στη θέση αυτή, με χρήση της εντολής ανάθεσης **fileReader = new Scanner(filename)**. Στη συνέχεια, καλώντας την **.populateURLsArraylist()**, αναθέτει στον πίνακα **myUrls[]** να αποθηκεύσει τα *host names* των urls του αρχείου.

Γ) Ανάλυση μεθόδων

Η κλάση αποτελείται από 3 μεθόδους, ως εξής:

- **private void populateURLsArraylist()**. Πραγματοποιεί validation στο **fileReader** με την εντολή **if (!fileReader.hasNextLine())**, κάτι το οποίο, επιτρέπει στον κώδικα να συνεχίσει, μόνο εφόσον το αρχείο έχει περιεχόμενο, δεν είναι δηλαδή άδειο. Στην περίπτωση που έχουμε άδειο αρχείο, ο χρήστης ενημερώνεται για αυτό με μήνυμα στο terminal και έχουμε το λεγόμενο *early return*. Αν το αρχείο έχει περιεχόμενο, γίνεται η προσπέλαση του μέχρι και την τελευταία γραμμή του, μέσα σε μία loop -> **while (fileReader.hasNextLine())**. Κατά τη διαδικασία αυτή και για κάθε επόμενη γραμμή, εντοπίζει τα forward slashes (" / "), στα οποία πραγματοποιεί **split** για τη συγκεκριμένη γραμμή, **fileReader.nextLine().split("/")** και αποθηκεύει όλα τα νέα κομμάτια της γραμμής, στον πίνακα αλφαριθμητικών *String[] splitUrl*. Συνεχίζοντας αναθέτει στον πίνακα **myUrls[]** να αποθηκεύει σε κάθε επανάληψη και για όσο διαρκεί η **while**, το αντίστοιχο *host name* των urls του αρχείου, το πρώτο δηλαδή τμήμα της κάθε γραμμής μέχρι την πλάγια κάθετο. Πιο συγκεκριμένα, αναφερόμαστε στο στοιχείο αυτό που αντιστοιχεί στη θέση μνήμης **splitUrl[0]**, για την κάθε γραμμή. Τέλος τερματίζουμε τη χρήση της *Scanner* καλώντας τη μέθοδο **fileReader.close()**.
- **public void removeDuplicates()**. Η συγκεκριμένη μέθοδος κάνοντας χρήση της κλάσης *ArrayList* παραλαμβάνει την **myUrls**, έτσι όπως έχει διαμορφωθεί από τον κατασκευαστή **public URLsVisited (File**

file), δημιουργεί έναν καινούριο πίνακα, τον οποίο αναθέτει εκ νέου στην μεταβλητή κλάσης **myUrls**, αφού πρώτα διαγράψει τα διπλότυπα περιεχόμενα και βάλει τα εναπομείναντα στοιχεία του πίνακα σε αλφαβητική σειρά. Το *unique sorting* το επιτυγχάνει η μέθοδος, δημιουργώντας αντικείμενο της κλάσης **TreeSet()**. Τέλος, θέτει την *boolean* μεταβλητή κλάσης **processedArraylist = true**, δηλαδή ότι είναι αληθές το γεγονός ότι η **myUrls** έχει υποστεί μετατροπή.

- **public void printAll()**. Η συγκεκριμένη μέθοδος, με τον τρόπο που καλείται από την *main()*, εκτελείται ανεξάρτητα με αν το η θέση αρχείου που δηλώνει ο χρήστης καταλήγει σε υπαρκτό αρχείο, γεμάτο ή άδειο. Για αυτό και δια να προβλεφθεί, η όποια μη επιθυμητή εκτύπωση κειμένου στο terminal, κάνει πρώτα validation της **myUrls** ελέγχοντας εάν ο πίνακας περιέχει στοιχεία ή είναι άδειος *if (! myUrls = null)*. Εφόσον δεν είναι άδειος ο πίνακας μέσα σε μία *for loop* γίνεται προσπέλαση του πίνακα και εκτυπώνονται κατά σειρά τα στοιχεία του:

```
for ( int i=0; i<myURLs.size(); i++ ) {  
    System.out.println( myURLs.get(i) );  
}
```

Η συγκεκριμένη for δύναται να αντικατασταθεί πλέον με την enhanced μορφή της:

```
for ( String url : myURLs ) {  
    System.out.println(url);  
}
```

Κάτι αρκετά ενδιαφέρον, είναι ότι βρήκα και δοκίμασα διάφορες εντολές, για να εκτυπώνονται τα στοιχεία του πίνακα, όπως την *myUrls.forEach(System.out : : println)* ή την *string.join("\n", myUrls)* κ.α, απλά η χρήση της *for..loop* που χρησιμοποίησα παραπάνω, μου φάνηκε ποιο ευανάγνωστη και πιο οικεία.

➤ Υλοποίηση URLsVisitedApp.java

Έγινε import των α) **java.io.FileNotFoundException** για να επιτευχθεί ο χειρισμός της συγκεκριμένης εξαίρεσης με *try..catch* στην **.fileAnalyzer()**, β) **java.util.Scanner** για να επιτρέψουμε στο πρόγραμμα να δέχεται καταχωρίσεις από τον user.

A) Ανάλυση μεθόδων

Η κλάση αποτελείται από 3 μεθόδους, ως εξής:

- **public static void main(String[] args)**. Η **main()** καλεί τη μέθοδο **.fileAnalyzer()** με παράμετρο το **filepath**, δηλαδή τη διαδρομή αρχείου (String) που επιστρέφει η **.getFilepath()**. Η **.fileAnalyzer()** με τη σειρά της καλεί τις μεθόδους της κλάσης **URLsVisited** και τρέχει ο κώδικας.
- **private static void fileAnalyzer(String filepath)**. Η μέθοδος δημιουργεί αντικείμενο της **URLsVisited**, μέσα σε μία *try..catch* για να διαχειριστεί την **FileNotFoundException**, με όνομα **urlsFile** και όρισμα τη

διαδρομή αρχείου που δίνεται ως παράμετρος. Στην περίπτωση που δε βρεθεί αρχείο γίνεται διαχείριση της exception, ενημερώνεται ο χρήστης με μήνυμα και πραγματοποιείται early return. Αν όμως έχει βρεθεί αρχείο, καλείται ο κατασκευαστής να διαβάσει το αρχείο και εφόσον υπάρχει περιεχόμενο, να εκτελέσει τις εντολές που αναφέραμε στην Ανάλυση Κατασκευαστή. Κατά σειρά καλούνται οι **file.printAll()**, **file.removeDuplicates()** και ξανά η **file.printAll()** για να εκτυπωθούν τα στοιχεία του πίνακα, πριν και μετά το unique sorting του περιεχομένου του.

- **private static String getFilepath()**. Η μέθοδος αυτή είναι υπεύθυνη να δημιουργήσει ένα αντικείμενο της κλάσης Scanner **keyboard**, για να διαβάσει από το terminal και ένα String **filepath** όπου εκεί θα αποθηκεύσει την καταχώριση του χρήστη, δηλαδή τη διαδρομή του αρχείου που επιθυμεί να αναλύσει το πρόγραμμα. Εκτυπώνονται τα αντίστοιχα μηνύματα στο terminal προς καθοδήγηση του χρήστη και τέλος επιστρέφει την διαδρομή αυτή, με τύπο επιστροφής String.

Θέμα 2: Επεξεργασία Αρχείων με URLs με Μενού επιλογών

Να γράψετε μια κλάση με όνομα **URLsVisitedMenu** που θα περιέχει τη μέθοδο **main()**. Το πρόγραμμα θα χρησιμοποιεί την κλάση **URLsVisited** του Θέματος 1. Θα εμφανίζει στην οθόνη το μενού επιλογών ως εξής:

```
1. Input the name of the file containing URLs you want to analyze
2. Input the path of the directory containing the files you to want to
   analyze for URLs
3. Exit
Which is your choice?
```

Το μενού θα εμφανίζεται μέσω κλήσης μιας μεθόδου **menu()** που θα περιέχεται στην κλάση **URLsVisitedMenu** και η οποία θα διαβάζει και θα επιστρέφει την επιλογή του χρήστη. Αυτή μπορεί να είναι μόνο 1, 2 ή 3 και μόνο τότε η **menu** θα επιστρέφει, διαφορετικά θα εμφανίζει πάλι τις επιλογές. Την περίπτωση ο χρήστης να εισάγει κάτι διαφορετικό από τους αριθμούς 1, 2 ή 3 να την ελέγξετε μέσω συνδυασμού χειρισμού εξαιρέσεων και μιας δομής επανάληψης **do... while** (φροντίστε το πρόγραμμά σας να αντιμετωπίζει την περίπτωση μη προβλεπόμενης εισόδου από το χρήστη όπως π.χ. ενός **String** ή ενός αριθμού **double** μέσω χειρισμού εξαιρέσεων).

Η κλάση **URLsVisitedMenu** θα περιέχει και τη μέθοδο **fileAnalyzer** όπως την κατασκευάσατε για το Θέμα 1 με την ίδια λειτουργικότητα.

Το πρόγραμμα θα εκτελείται επαναληπτικά (και εδώ μπορείτε να χρησιμοποιήσετε το **do... while**) μέχρις ότου ο χρήστης δώσει το 3 που το τερματίζει.

Αν ο χρήστης επιλέξει 1, το πρόγραμμα:

- Θα ζητά και θα διαβάζει από το πληκτρολόγιο το μονοπάτι και όνομα ενός αρχείου (ως ενιαίο **String**) του οποίου τα περιεχόμενα θα αναλυθούν για την εύρεση των κόμβων που περιέχουν.
- Στη συνέχεια θα καλεί τη μέθοδο **fileAnalyzer** με όρισμα το **String** που περιέχει το μονοπάτι και αρχείο που δόθηκαν. Η μέθοδος αυτή θα τυπώνει στην οθόνη αρχικά τους κόμβους που περιέχονται στο αρχείο εισόδου και στη συνέχεια τους μοναδικούς κόμβους ταξινομημένους.

Αν ο χρήστης επιλέξει 2, το πρόγραμμα:

- Θα ζητά και θα διαβάζει από το πληκτρολόγιο το μονοπάτι στο οποίο περιέχονται τα αρχεία που θα αναλυθούν για την εύρεση των κόμβων που περιέχουν.
- Θα ελέγχει αν το μονοπάτι υπάρχει και στη συνέχεια θα διαβάζει τα περιεχόμενα του φακέλου εισόδου (τα ονόματα των αρχείων) σε έναν πίνακα συμβολοσειρών με όνομα **filenames**.
- Για κάθε στοιχείο του πίνακα **filenames** θα καλεί τη μέθοδο **fileAnalyzer**. Αυτή και πάλι θα τυπώνει στην οθόνη αρχικά τους κόμβους που περιέχονται στο αρχείο εισόδου και στη συνέχεια τους μοναδικούς κόμβους ταξινομημένους.

Αν ο χρήστης επιλέξει 3, το πρόγραμμα

- Θα τερματίζει

➤ Υλοποίηση URLsVisitedMenu.java

Το πρόγραμμα αποτελείται από μία κλάση μέσα στην οποία βρίσκεται και η **main()** μέθοδος. Κάνουμε **import** τις α) **thema1.URLsVisited**, β) **java.util.Scanner** για να επιτρέψουμε στο πρόγραμμα να δέχεται καταχωρίσεις από τον **user**, γ) **java.io.File** για να δημιουργήσουμε ένα νέο αντικείμενο της κλάσης αυτής, δ) **java.util**.

InputMismatchException για να επιτευχθεί ο χειρισμός της συγκεκριμένης εξαίρεσης με try catch, όταν δοθεί άλλη καταχώριση από τον χρήστη μη αποδεκτή.

A) Μεταβλητές Κλάσης

Η 1 στατικές μεταβλητή που δηλώθηκε είναι η εξής:

- **private static final Scanner keyboard = new Scanner(System.in)**, στην οποία αποθηκεύεται η καταχώριση του user.

B) Ανάλυση μεθόδων

Η κλάση αποτελείται 03 μεθόδους, ως εξής:

- **public static void main(String[] args)**. Η main() έχει δύο τοπικές μεταβλητές, την **String directory** στην οποία αποθηκεύεται η διαδρομή αρχείου που θα πληκτρολογήσει ο χρήστης και την **int menuchoice** στην οποία θα αποθηκεύεται η καταχώριση του χρήστη εκ των επιλογών 1,2 ή 3. Αυτό γίνεται καλώντας την menu() η οποία επιστρέφει την καταχώριση του χρήστη εντός μίας do_while με statement **menuchoice != 3**, έως ότου δηλαδή ο χρήστης ζητήσει τον τερματισμό του προγράμματος πληκτρολογώντας το 3.
- **private static int menu()**. Η συγκεκριμένη μέθοδος προβάλλει στην οθόνη τις βασικές ερωτήσεις του προγράμματος, σύμφωνα με την εκφώνηση της άσκησης, απαιτώντας να καταχωρισθεί ένας integer μεταξύ των 1,2 ή 3 και επιστρέφει το αριθμητικό input του user. Εντός μίας do..while loop, με τη χρήση μίας try..catch, γίνεται ο χειρισμός της εξαίρεσης InputMismatchException(), η οποία δύναται να εκδηλωθεί με οποιαδήποτε άλλη καταχώριση του χρήστη **if(input<1 || input>3) { throw new InputMismatchException();}**.
- **private static void fileAnalyzer(String directory)**. Η μέθοδος δημιουργεί αντικείμενο της **URLsVisited**, μέσα σε μία try..catch για να διαχειριστεί την **FileNotFoundException**, με όνομα **urlsFile** και όρισμα τη διαδρομή αρχείου που δίνεται ως παράμετρος. Στην περίπτωση που δε βρεθεί αρχείο γίνεται διαχείριση της exception, ενημερώνεται ο χρήστης με μήνυμα και πραγματοποιείται early return. Αν όμως έχει βρεθεί αρχείο, καλείται ο κατασκευαστής να διαβάσει το αρχείο και εφόσον υπάρχει περιεχόμενο, να εκτελέσει τις εντολές που αναφέραμε στην Ανάλυση Κατασκευαστή του Θέματος 1. Κατά σειρά καλούνται οι **file.printAll()**, **file.removeDuplicates()** και ξανά η **file.printAll()** για να εκτυπωθούν τα στοιχεία του πίνακα, πριν και μετά το unique sorting του περιεχομένου του.

Θέμα 3: Πωλήσεις Supermarket

Μια αλυσίδα Supermarket θέλει να καταγράψει για τα καταστήματα που διαθέτει, τις πωλήσεις των προϊόντων που πραγματοποίησαν. Η αλυσίδα Supermarket διαθέτει καταστήματα σε διάφορες πόλεις και όλα εμπορεύονται τα ίδια προϊόντα. Για τους σκοπούς της άσκησης να γράψετε μια κλάση με όνομα `SupermarketSales` η οποία θα περιέχει ως μέλη:

- Έναν public πίνακα δύο διαστάσεων με όνομα **sales** και με στοιχεία δεκαδικούς αριθμούς (double). Σε κάθε γραμμή του πίνακα θα αποθηκεύονται οι πωλήσεις (σε ευρώ) ενός καταστήματος της αλυσίδας supermarket για όλα τα προϊόντα.
- Δύο public ακέραιους **katastimata** και **products** όπου αντίστοιχα θα αποθηκεύεται το πλήθος των καταστημάτων που διαθέτει η αλυσίδα Supermarket και το πλήθος των προϊόντων που διαθέτει η αλυσίδα. Το πλήθος των καταστημάτων αντιστοιχεί στο πλήθος των γραμμών του πίνακα sales και το πλήθος των προϊόντων στις στήλες του πίνακα αντίστοιχα.
- Έναν κατασκευαστή ο οποίος θα δέχεται ως όρισμα έναν πίνακα δύο διαστάσεων. Θα ελέγχει αν τα περιεχόμενα στοιχεία του πίνακα είναι έγκυρες τιμές πωλήσεων προϊόντων (≥ 0). Στην περίπτωση που δεν είναι, θα εμφανίζει σχετικό μήνυμα λάθους και το πρόγραμμα θα τερματίζει (μπορείτε να χρησιμοποιήσετε τη μέθοδο `System.exit`). Αν τα δεδομένα είναι έγκυρα, θα αναθέτει τον πίνακα αυτόν στον **sales**, και θα αποθηκεύει στην **katastimata** το πλήθος των γραμμών του και στην **products** το πλήθος των στηλών.
- Μια δημόσια μέθοδο **getKatastimataSales** που θα διατρέχει τον πίνακα και θα υπολογίζει για το κάθε κατάστημα το άθροισμα των πωλήσεων και θα το εμφανίζει στην οθόνη.
- Μια δημόσια μέθοδο **getProductSales** που θα διατρέχει τον πίνακα, και θα υπολογίζει για το κάθε προϊόν το άθροισμα των πωλήσεων καθώς και τον μέσο όρο των πωλήσεων ανα κατάστημα και θα το εμφανίζει στην οθόνη.
- Μια δημόσια μέθοδο **getSupermarketSales** που θα διατρέχει τον πίνακα, και θα υπολογίζει το άθροισμα των πωλήσεων της αλυσίδας σουπερμαρκετ και τον μέσο όρο των πωλήσεων ανα κατάστημα και θα το εμφανίζει στην οθόνη.

Για να δοκιμάσετε την κλάση σας, να γράψετε πρόγραμμα που θα δημιουργεί ένα αντικείμενο της κλάσης περνώντας της τον πίνακα για τις πωλήσεις τριών καταστημάτων και τεσσάρων προϊόντων, σύμφωνα με τα δεδομένα που βλέπετε στην Εικόνα 1. Το πρόγραμμα που θα περιέχει τη `main`, θα βρίσκεται σε μια δεύτερη κλάση με όνομα `MainProgramSupermarket`.

Για την επίλυση της άσκησης δημιουργήθηκαν δύο κλάσεις η `public class SupermarketSales` (`SupermarketSales.java`) και η `public class MainProgramSupermarket` (`MainProgramSupermarket.java`) στην οποία βρίσκεται η `main()` μέθοδος.

➤ Υλοποίηση SupermarketSales.java

A) Μεταβλητές Κλάσης

Οι 3 μεταβλητές που δηλώθηκαν είναι οι εξής:

- **public int katastimata** = 0; Ο ακέραιος στον οποίο θα αντιστοιχήσουμε το πλήθος των καταστημάτων που διαθέτει η αλυσίδα Supermarket. Αρχικοποιήθηκε σε 0.

- **public** *int products* = 0; Ο ακέραιος στον οποίο θα αντιστοιχήσουμε το πλήθος των προϊόντων που διαθέτει η αλυσίδα. Αρχικοποιήθηκε σε 0.
- **public** *double[][] sales*; Ένας public πίνακα δύο διαστάσεων, με στοιχεία δεκαδικούς αριθμούς (double). Σε κάθε γραμμή του πίνακα θα αποθηκεύονται οι πωλήσεις ενός καταστήματος της αλυσίδας supermarket για όλα τα προϊόντα. Σε κάθε στήλη του πίνακα θα αποθηκεύονται οι πωλήσεις κάθε ενός προϊόντος.

B) Ανάλυση Κατασκευαστή

Η κλάση αποτελείται από 1 κατασκευαστή, ως εξής:

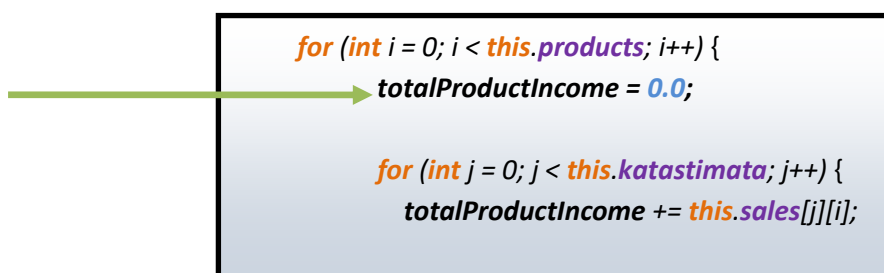
- **public** *SupermarketSales(double[][] sales)*. Ο κατασκευαστής δέχεται ως όρισμα έναν πίνακα δύο διαστάσεων. Αναθέτει τον πίνακα αυτόν στον **sales**, και αποθηκεύει στην **katastimata** το πλήθος των γραμμών του και στην **products** το πλήθος των στηλών. Ύστερα κάνει προσπέλαση του δοσμένου πίνακα μέσα σε δύο for..loops και στην περίπτωση που κάποιο από τα δεδομένα δεν είναι έγκυρο, δηλαδή ικανοποιείται η if (sales[i][j]< 0), θα εμφανίζει σχετικό μήνυμα λάθους και το πρόγραμμα θα τερματίζει με την εντολή *System.exit(0)*;

Γ) Ανάλυση μεθόδων

Η κλάση αποτελείται 03 μεθόδους, ως εξής:

- **public** *void getKatastimataSales()*. Η μέθοδος διατρέχει τον πίνακα και υπολογίζει για το κάθε κατάσταση το άθροισμα των πωλήσεων, το οποίο αναθέτει στην τοπική μεταβλητή *double totalKatastimalIncome* και το εμφανίζει στην οθόνη με κατάλληλο μήνυμα. Η πράξη γίνεται μέσα σε δύο for..loop.
- **public** *void getProductsSales()*. Η μέθοδος διατρέχει τον πίνακα και υπολογίζει για το κάθε προϊόν το άθροισμα των πωλήσεων, το οποίο αναθέτει στην τοπική μεταβλητή *double totalProductIncome*, καθώς και τον μέσο όρο των πωλήσεων ανά κατάσταση, τον οποίο αναθέτει στην τοπική μεταβλητή *double averageProductIncome* και τα εμφανίζει στην οθόνη με κατάλληλο μήνυμα. Η πράξη γίνεται μέσα σε δύο for..loop.
- **public** *void getSupermarketSales()*. Η μέθοδος διατρέχει τον πίνακα και υπολογίζει το άθροισμα των πωλήσεων της αλυσίδας σουπερμάρκετ, το οποίο αναθέτει στην τοπική μεταβλητή *double totalChainIncome* και τον μέσο όρο των πωλήσεων ανά κατάσταση και το εμφανίζει στην οθόνη με κατάλληλο μήνυμα. Η πράξη γίνεται μέσα σε δύο for..loop.

Να τονίσω σε αυτό το σημείο, κάτι σημαντικό για τις μεθόδους *.getKatastimataSales()*, *.getProductsSales()*. Για να έχουμε το επιθυμητό αποτέλεσμα στις πράξεις μας, πρέπει να γίνεται η αρχικοποίηση σε 0.0 των προς εκτύπωση μεταβλητών, σε κάθε επανάληψη for..loop όπως για παράδειγμα φαίνεται παρακάτω:



```

for (int i = 0; i < this.products; i++) {
    totalProductIncome = 0.0;

    for (int j = 0; j < this.katastimata; j++) {
        totalProductIncome += this.sales[j][i];
    }
}

```

The diagram shows a light blue rectangular box containing the above code. A green arrow points from the left towards the initialization line `totalProductIncome = 0.0;`.

➤ Υλοποίηση MainSupermarketSales.java

Σε αυτήν την κλάση βρίσκεται η *public static void main(String[] args)*. Μέσα στη *main()* δημιουργούμε ένα αντικείμενο της κλάσης *SupermarketSales* σύμφωνα με το ζητούμενο της άσκησης με όνομα **incomes** και δίνουμε ως παράμετρο τον δισδιάστατο πίνακα, που έχουμε προηγουμένως αναθέσει hardcoded στην τοπική μεταβλητή **double[][] sales**. Ενημερωτικά τα δεδομένα του πίνακα είναι ίδια με του παραδείγματος. Κατά σειρά καλούνται οι **.getKatastimataSales()**, **.getProductsSales()** και **.getSupermarketSales()** μέθοδοι, για να εκτυπωθούν τα αθροίσματα και οι μέσοι όροι που υπολογίστηκαν από τον δοσμένο πίνακα.

Μέσα στη main():



```
double[][] sales = { ..... };  
  
SupermarketSales incomes = new SupermarketSales(sales);  
  
incomes.getKatastimataSales();  
  
incomes.getProductsSales();  
  
incomes.getSupermarketSales();
```

		Πωλήσεις			
		0	1	2	3
Καταστήματα	0	12502,5	2506,75	8088,33	1289,55
	1	10085,65	1505,85	5800,65	1184,55
	2	19258	1158,57	9871,22	1289,78

Εικόνα 1. Πίνακας δεδομένων

```
Katastima no:0  
Total income is 24387.13  
Katastima no:1  
Total income is 18576.7  
Katastima no:2  
Total income is 31577.57
```

(α)

```
Proion no:0  
Total income is 41846.15  
Average income ana katastima is 13948.716666666667  
Proion no:1  
Total income is 5171.17  
Average income ana katastima is 1723.7233333333334  
Proion no:2  
Total income is 23760.199999999997  
Average income ana katastima is 7920.066666666666  
Proion no:3  
Total income is 3763.88  
Average income ana katastima is 1254.6266666666668
```

(β)

```
The total income of all the Supermarket's katastimata is 74541.4  
Average income per katastima is 24847.133333333333
```

(γ)

Εικόνα 2. Παράδειγμα αναλύσεων και εμφανίσεων στοιχείων στον χρήστη

Θέμα 4: Επέκταση Κλάσης – Πωλήσεις Supermarket σε αρχείο

Να επεκτείνετε την κλάση **SupermarketSales**, δημιουργώντας μια υποκλάση με όνομα **SupermarketSalesToFile**. Η υποκλάση θα περιλαμβάνει λειτουργικότητα με την οποία θα αποθηκεύει τα αποτελέσματα των πωλήσεων σε αρχείο κειμένου στο δίσκο, αντί να τα εμφανίζει στην οθόνη.

Στην υποκλάση **SupermarketSalesToFile** θα προσθέσετε:

- Ένα δημόσιο πεδίο σε επίπεδο κλάσης (static) τύπου αλφαριθμητικού με όνομα **supermarket name**, που θα αποθηκεύει το όνομα της αλυσίδας Supermarket.
- Ένα ιδιωτικό αλφαριθμητικό πεδίο με όνομα **poli**, όπου θα αποθηκεύεται η Πόλη στην οποία υπάρχει κατάσταση ή καταστήματα.
- Έναν κατασκευαστή ο οποίος θα δέχεται ως όρισμα έναν πίνακα δύο διαστάσεων και θα χρησιμοποιεί για αρχικοποίηση τον κατασκευαστή της υπερκλάσης (με χρήση του **super**) με την ίδια παράμετρο. Δεν χρειάζεται να δηλώσετε πάλι τα **rows**, **columns**, **grades** καθώς κληρονομούνται από την υπερκλάση.
- Μια δημόσια static void μέθοδο με όνομα **setSupermarketName** η οποία θα δέχεται ως όρισμα ένα αλφαριθμητικό που είναι το όνομα της αλυσίδας Supermarket και θα το αποθηκεύει στην στατική μεταβλητή.
- Μια δημόσια void μέθοδο με όνομα **setPoli** η οποία θα δέχεται ως όρισμα ένα αλφαριθμητικό που είναι η πόλη και θα το αποθηκεύει στη μεταβλητή **poli**.
- Μια δημόσια String μέθοδο με όνομα **getPoli** που θα επιστρέφει την πόλη .
- Επιπλέον, να υπερκαλύψετε (override) τη μέθοδο **getKatastimaSales**, ώστε αυτή να υπολογίζει τα αποτελέσματα (άθροισμα πωλήσεων) αλλά αντί για την οθόνη, να τα αποθηκεύει σε ένα αρχείο κειμένου στον δίσκο με όνομα **<katastima>.txt** όπου **<katastima>** είναι το όνομα του αντίστοιχου καταστήματος, π.χ. **athina.txt** ή **thessaloniki.txt**, **patra.txt** κτλ. Δείγμα της μορφής του αρχείου μπορείτε να δείτε στην Εικόνα 3.

Για να δοκιμάσετε την κλάση σας, να γράψετε πρόγραμμα (σε μια δεύτερη κλάση με όνομα **MainProgram4**) που θα δημιουργεί δύο αντικείμενα **SupermarketSalesToFile** με τις πωλήσεις 3 καταστημάτων της πόλης **athina** και 2 καταστημάτων της πόλης **patra** αντίστοιχα και θα αποθηκεύει τα αποτελέσματα σε δύο ξεχωριστά αρχεία. Οι πωλήσεις είναι για 4 προϊόντα για κάθε κατάσταση και μπορείτε να ορίσετε τυχαίες τιμές που επιθυμείτε στις αντίστοιχες μεταβλητές σας για τις πωλήσεις. Δεν απαιτείται να γίνεται εισαγωγή τους από τον χρήστη από το πληκτρολόγιο.

- Το πρόγραμμά σας αρχικά θα καλεί τη στατική μέθοδο **setSupermarketName()** για να καθορίσει το όνομα του supermarket σε "Sklavenitis". Σημειώστε, ότι η μέθοδος αυτή μπορεί να κληθεί άμεσα, χωρίς να έχουμε νωρίτερα δημιουργήσει αντικείμενο, οπότε η κλήση θα γίνεται με το όνομα της κλάσης, δηλαδή **SupermarketSalesToFile.setSupermarketName ("Sklavenitis")**.
- Στη συνέχεια θα δημιουργεί τα δύο αντικείμενα της κλάσης, ένα για κάθε **poli**, και θα δίνει τις κατάλληλες τιμές στο πεδίο **poli** του κάθε αντικειμένου.
- Τέλος, θα καλεί τη μέθοδο **getkatastimataSales** των δύο αντικειμένων για να αποθηκεύει τα αποτελέσματα στα αντίστοιχα αρχεία.

Για την επίλυση της άσκησης δημιουργήθηκαν δύο κλάσεις η **public class SupermarketSalesToFile** (**SupermarketSalesToFile.java**) και η **public class MainProgram4** (**MainProgram4.java**) στην οποία βρίσκεται η **main()** μέθοδος.

➤ Υλοποίηση SupermarketSalesToFile.java

Έγινε import των α) *thema3.SupermarketSales* και β) *java.io.** για να εκμεταλλευτούμε την κλάση *FileWriter* και την εξαίρεση *IOException* την οποία ενδέχεται να προκαλέσει η χρήση της *FileWriter*.

A) Μεταβλητές Κλάσης

Οι 3 μεταβλητές που δηλώθηκαν είναι οι εξής:

- **public static String supermarketName;**, στην οποία θα αποθηκεύεται το όνομα της αλυσίδας Supermarket.
- **private String poli;**, στην οποία θα αποθηκεύεται η Πόλη στην οποία υπάρχει κατάστημα ή καταστήματα.

B) Ανάλυση Κατασκευαστή

Η κλάση αποτελείται από 1 κατασκευαστή, ως εξής:

- **public SupermarketSalesToFile (double[][] sales).** Ο κατασκευαστής δέχεται ως όρισμα έναν πίνακα δύο διαστάσεων και χρησιμοποιεί για αρχικοποίηση τον κατασκευαστή της υπερκλάσης *SupermarketSales* (με χρήση του **super**) με την ίδια παράμετρο.

Γ) Ανάλυση μεθόδων

Η κλάση αποτελείται 03 μεθόδους, ως εξής:

- **public static void setSupermarketName(String name).** Η μέθοδος δέχεται ως όρισμα ένα αλφαριθμητικό *name*, το οποίο αντιστοιχεί στο όνομα της αλυσίδας Supermarket και το αποθηκεύει στην στατική μεταβλητή κλάσης *supermarketName*.
- **public void setPoli(String poli).** Η μέθοδος δέχεται ως όρισμα ένα αλφαριθμητικό που αντιστοιχεί στην πόλη που βρίσκονται τα καταστήματα της αλυσίδας και το αποθηκεύει στη μεταβλητή κλάσης *poli*.
- **public String getPoli().** Η μέθοδος επιστρέφει τη μεταβλητή *poli*, δηλαδή το όνομα της πόλης.
- **public void getKatastimataSales().** Η μέθοδος υπερκαλύπτει τη μέθοδο *getKatastimaSales* της υπερκλάσης *SupermarketSales* και υπολογίζει το άθροισμα πωλήσεων ανά κατάστημα, αλλά αντί για την οθόνη, τα αποθηκεύει σε ένα αρχείο κειμένου στον δίσκο και συγκεκριμένα στο φάκελο που εμπεριέχει τα *thema3* και *thema4*, της *Exercise2*. Κάτι που είναι σημαντικό να επισημανθεί είναι, ότι οι υπολογισμοί, καθώς και η αποθήκευση τους σε αρχείο, γίνονται εντός μίας *try..catch*, ώστε να επιτευχθεί ο χειρισμός της εξαίρεσης *IOException*, η οποία είναι δυνατόν να προκληθεί από τη χρήση της *FileWriter*. Πιο αναλυτικά δημιουργούμε ένα αντικείμενο της *FileWriter*, η οποία επεκτείνει την *OutputStreamWriter*, με όνομα *file*, *FileWriter file = new FileWriter (getPoli() + ".txt")*; Η εντολή αυτή δημιουργεί ένα αρχείο με όνομα, το όνομα της πόλης, που θα επιστρέψει η μέθοδος *.getPoli()* και κατάληξη αρχείου *.txt*. Χρησιμοποιώντας τη μέθοδο *.write()* της *FileWriter* γίνεται εγγραφή, όποιου αλφαριθμητικού String δοθεί ως όρισμα, στο αρχείο που δημιουργήθηκε νωρίτερα. Εκτελείται η προσπέλαση του πίνακα που δίνεται στο πρόγραμμα και τα αποτελέσματα των υπολογισμών εγγράφονται στο αρχείο. Τέλος σταματάει η καταγραφή και η χρήση της *FileWriter*, καλώντας τη μέθοδο *.close()*. Δείγμα της μορφής του αρχείου μπορούμε να δούμε στην Εικόνα 3 στην επόμενη σελίδα.

Athina.txt	Patra.txt
1 Sklavenitis	1 Sklavenitis
2 Poli:Athina	2 Poli:Patra
3 katastima no 0	3 katastima no 0
4 Total income is 24387.13	4 Total income is 17506.25
5 katastima no 1	5 katastima no 1
6 Total income is 18576.7	6 Total income is 30777.04
7 katastima no 2	7
8 Total income is 31577.57	
9	

Εικόνα 3. Παράδειγμα αναλύσεων και εμφανίσεων στοιχείων στον χρήστη

➤ Υλοποίηση MainProgram4.java

Σε αυτήν την κλάση βρίσκεται η `public static void main(String[] args)`. Μέσα στη `main()` πρώτα καλούμε την `.setSupermarketName()` και ως παράμετρο δίνουμε το όνομα της αλυσίδας σούπερμαρκετ. Δημιουργούμε συνολικά δύο αντικείμενα της κλάσης **SupermarketSalesToFile** σύμφωνα με το ζητούμενο της άσκησης με ονόματα **store1** και **store2** και δίνουμε ως παράμετρο δισδιάστατους πίνακες, που έχουμε προηγουμένως αναθέσει hardcoded στις τοπικές μεταβλητές `double[][] branchThessalonikiSales` και `double[][] branchPatraSales` αντίστοιχα. Κατά σειρά καλούνται οι `.setPoli()` και `.getKatastimataSales()`, για να πάρουμε το αντίστοιχο όνομα που θα δοθεί στο αρχείο .txt και να πραγματοποιηθούν οι υπολογισμοί των αθροισμάτων πωλήσεων ανά κατάσταση και να καταγραφούν στο αρχείο .txt.

```
SupermarketSalesToFile.setSupermarketName("Sklavenitis");
```

```
double[][] branchThessalonikiSales = {{12502.5, 2506.75, 8088.33, 1289.55}, {10085.65, 1505.85, 5800.65, 1184.55}, {19258, 1158.57, 9871.22, 1289.78}};
```

```
SupermarketSalesToFile stores1 = new SupermarketSalesToFile(branchThessalonikiSales);
```

```
stores1.setPoli("Thessaloniki");
```

```
stores1.getKatastimataSales();
```

```
double[][] branchPatraSales = {{10502.5, 2406.75, 8188.33, 1489.55}, {10585.65, 1805.85, 4800.65, 1584.55}};
```

```
SupermarketSalesToFile stores2 = new SupermarketSalesToFile(branchPatraSales);
```

```
stores2.setPoli("Patra");
```

```
stores2.getKatastimataSales();
```